Abstract In this paper, we present an efficient yet easy-to-implement technique which performs automatic skinning and animation of skeletal models. At a pre-processing stage, a character model is firstly decomposed into a number of segments per bone basis, and each segment is then subdivided into several chunks. A convex cage is automatically created for each chunk. The skinning and animation of skeletal models is achieved via two steps. At the first step, by minimizing a sum of several energy terms, chunk cages are implicitly skinned to the skeleton and animated. These energies are carefully designed to prevent unnatural volume change and guarantee smooth deformation transition between adjacent cages. At the second step, the model mesh vertices, represented as the mean-value coordinates with reference to proper cage vertices, are updated via cage-based deformation technique. Our approach avoids the labor-intensive process of vertex weighting and cage generation. Given the motion of a skeleton, the character model can be animated automatically.

Keywords Skeleton · Cage · Deformation

1 Introduction

1.1 Motivation

Animating and deforming articulated 3D models have attracted keen interest in the movie industry and video game industry. A model is called articulated or skeletal if its general poses can be represented by a simple description as a skeleton. In that sense most characters in the digital entertainment products, including human models, are articulated.

The skeleton-driven deformation technique (also called vertex weighting, linear skin blending or skeleton subspace deformation [7, 23]) has been a popular method in animating articulated models in industry. However, it is less effective in preserving local surface details and is prone to cause candy-wrapping and elbow-collapse artifacts. Careful vertex weight-tuning is required to avoid visual artifacts, but this process is a piece of notoriously labor-intensive work.

Cage-based technique is a way of deforming generic meshes, not necessarily skeletal models. A cage is a low polygon-count closed polyhedron representing the silhouette of an object model. When the shape of the cage changes, the enclosed mesh is updated accordingly. The disadvantage is that a cage must be pre-built by the user. Elaborated deformations are possible only with carefully designed cages, yet creating a proper cage is not trivial. Moreover, in most existing literatures, cages are edited by manipulating the vertices directly which makes it difficult to get the intended target cage shapes, even for a cage of only a few hundred faces.

We envision a system that eliminates the tedium of vertex weighting and exploits the flexibility of cages. To support this functionality, we need a method (as shown in Fig. 1) that takes a character, a skeleton, and a motion of that skeleton as input, and outputs the moving character. The missing portion is the skinning. In this paper, we propose a skinning technique...
technique that combines the skeleton-driven and the caged-based deformation. Our method avoids the labor-intensive process of vertex weighting and cage generation, while preserving local details well in the deformation results.

1.2 Overview and contributions

In our proposed technique, a pre-processing sets up the automatic skinning so that proper deformation occurs at runtime. At the pre-processing stage, a character model is decomposed into a number of segments per bone basis, and each segment is then subdivided into several chunk meshes. A convex cage is automatically created for each chunk, and we name it chunk cage. The skinning and animation of skeletal models is achieved via two steps. At the first step, by minimizing a sum of several types of energies, chunk cages are implicitly skinned to the skeleton and animated. These energies are carefully designed to prevent unnatural volume change and guarantee smooth deformation transition between adjacent cages. At the second step, each cage transfers its deformation to the enclosed mesh via cage-based technique. The proposed technique has several highlights:

1. Rather than explicitly weighting the influence of each bone upon the mesh vertices as in most skeleton-driven methods, our approach automatically and implicitly binds a mesh onto the skeleton.
2. Rather than altering each mesh vertex independently in typical vertex-weighting methods, our approach updates the mesh vertices by minimizing an energy function, taking account the influences of the neighborhood as well as the volumetric constraints, thus yields smooth deformation results with the model volume well preserved.
3. Rather than deforming an object model by a single cage which usually has to be designed carefully and manually, our approach controls the object model by a serial of chunk cages which are created automatically. Each chunk cage controls a portion of the object model, which makes it flexible in controlling articulated models.

The rest of the paper is structured as follows. After reviewing the related work in Sect. 2, some background knowledge are introduced in Sect. 3. The method for constructing chunk cages is proposed in Sect. 4. The method for automatic skinning and deforming the object model is proposed in Sect. 5. Experimental results and analysis are given in Sect. 6. The whole paper is concluded in Sect. 7.

2 Related work

There are abundant literatures on model deformation and model animation. In this section, we only give an overview of those which are most related to our work.

Most skeleton-based deformation methods are based on linear vertex weighting, including those implemented in commercial packages such as Maya, 3DS Max [26]. In these packages, the vertex weighting is left to the user to set up the weights manually. Such work is time-consuming, and usually can only be accomplished by professional animators. Researchers have been working toward automatic vertex-weighting methods to alleviate human labor. Among them, heat diffusion based technique is an increasingly popular solution [3, 36] in which the influential weight of a bone upon the mesh vertices are allocated by solving a heat equilibrium equation.

Intrinsically, as this type of methods use simple linear technique to simulate the inherently non-linear relationship between the skin and the skeleton, unnatural deformation would unavoidably appear in some cases. To address this issue, a variety of methods have been proposed to enhance the realism such as adding additional joints [29], spherical blend skinning [21], quaternion blending methods [22], sweep-based methods [16], curved-skeleton-based methods [13, 38], and moving-least-squares-based methods [11]. In summary, though these methods perform well in certain particular situations, in most of them, each vertex update is totally controlled by the skeleton information, ignoring the influence of its adjacent vertices, which is not helpful for preserving the geometric local details in deformation.

Gradient domain based mesh deformation (refer to [42] for a survey) such as Laplacian Mesh Deformation (LMD) methods [32, 33] or Poisson shape editing methods [40] achieve the local details preservation with the help of differential geometry. However, these methods may introduce unnatural shearing and scaling. To address this issue, Sorkine et al. [34] proposed an “as-rigid-as-possible” deformation method which seeks to keep the local transformation as-rigid-as-possible across the mesh surface.

Gradient domain based method preserves local surface details well, however, it can result in a volume loss, if there
is no volumetric constraint added. To address this issue, Zhang et al. [41] added sectional mesh constraints into the framework of Sorkine et al.’s [34] to reduce the volume change in the deformation results. Similar constraints are also used in [10]. Zhou et al. extend the LMD to the volumetric domain [43]; their method can prevent unnatural volume changes. Huang et al. further developed this technique to precisely preserve the model volume [15].

While gradient domain based methods mostly focus on surface deformation, cage-based methods aim at volume deformation using either mean-value [12, 19], harmonic [5, 18], or Green coordinates [25]. If the cage is concave, the mean-value coordinate of a spatial point inside the cage can be negative, causing the point moving in the opposite direction to the cage vertex in deformation. The harmonic coordinates [18] was proposed to address this issue. They are non-negative and do not possess a local extremum; thus they are more intuitive in controlling the deformation than the mean-value coordinates method. However, compared to mean-value coordinates, harmonic coordinates do not hold closed-form formulas, and computing the barycentric coordinates for a convex cage is more costly. Different from mean-value and harmonic coordinates which solely depend on cage vertices, Green coordinates [25] depend on both the vertex positions and face normal of a cage and usually yield pleasing results. Since face normal is computed from a cross-product of certain cage vertices, a spatial point is non-linear combination of the cage vertices, thus non-linear equations have to be solved. Due to the fact that the mean-value method only needs to solve linear equations, we adopt convex cages together with the mean-value method. Convex cages do not suffer the problems that concave cages have.

A few researchers have proposed solutions for automatic generation of cages for 3D models. In the mesh voxelization based method [37], the cage is created by smoothing a closed mesh which is composed of the outer faces of the voxels intersecting the input model. Their method creates effective cages in most cases. However, the topology of the resulting cage relies on voxel resolution. Ben-Chen et al. [6] proposed a heuristic method for automatic cage generation by iteratively enveloping the input shape, simplifying the envelope, and offsetting the simplified mesh. Their method suffers similar problem as [37], i.e. when two parts are very close in Euclidean distance while being far from each other in surface geodesic distance, in the resulting cage, these two parts might be fused. In the work of Ju et al. [20], a set of skinning templates are pre-defined by using cage-based deformations. Each skinning template relates to a proper cage template. The cage for an object character model can be automatically composed with the pre-built cage templates followed by interactive editing. Thus the skinning templates can be reusable on the object character model.

Other deformation methods include physically-based methods [9, 14, 35] and example-based methods [17, 28–30, 36]. These methods, either simulating deformations physically or statistically, can produce plausibly realistic results; however, most of them either require huge computation or need a large database of deformation examples.

### 3 Background knowledge

A mesh \( M = (V, E, F) \) is represented by a set of vertices \( V = \{v_i \in \mathbb{R}^3 \} \), a set of triangles \( F = \{f_i\} \) and a set of edges \( E = \{e_i\} \), where \( f_i = (v_{i,0}, v_{i,1}, v_{i,2}) \) and \( e_i = (v_{i,0}, v_{i,1}) \) define the geometry and topology of the surface. The number of \( V \) on \( M \) is \( N^M_V \). We use bold letter to denote the coordinate vector of a point, e.g. \( v_i \) for \( v_i \).

#### 3.1 Laplacian mesh deformation

The Laplacian of \( M \) computes the difference between each vertex \( v_i \) and a linear combination of its neighboring points [32, 33, 43]:

\[
\delta_i = L(v_i) = v_i - \sum_{j \in N(i)} w_{ij} v_j, \quad \sum_{j \in N(i)} w_{ij} = 1 \tag{1}
\]

where \( L(v_i) \) is the Laplacian operator of the vertex \( v_i \), \( \delta_i \) is the Laplacian coordinate of \( v_i \), \( N(i) = \{j | (i, j) \in E\} \) are the number of adjacent edges on vertex \( v_i \). Weights \( w_{ij} \) can chosen as cotangent functions [27]:

\[
w_{ij} \propto \cot \alpha_{ij} + \cot \beta_{ij}
\]

where \( \alpha_{ij} \) and \( \beta_{ij} \) are the two angles opposite to the edge \( v_i v_j \) in the two triangles sharing this edge.

When mesh \( M \) is deformed, its Laplacian deformation energy \( E_L \) could be represented as

\[
E_L = \sum_{i=0}^{N^M_V} \| L(v'_i) - \delta'_i \|^2 \tag{2}
\]

where \( v'_i \) is the deformed \( v_i \), \( \delta'_i \) is the Laplacian coordinate of \( v'_i \) and can be computed as

\[
\delta'_i = T_i \delta_i \tag{3}
\]

where \( T_i \) is a 3-by-3 matrix which transforms \( v_i \) to \( v'_i \).

#### 3.2 Barycentric coordinate

Suppose a point \( q_i \) lies inside a triangle \( f_j \), then its position can be represented by its barycentric coordinate \((w_0, w_1, w_2)\) on \( f_j \) as

\[
q_i = w_0 v_{j,0} + w_1 v_{j,1} + w_2 v_{j,2}, \quad w_0 + w_1 + w_2 = 1 \tag{4}
\]
We follow the mean-value coordinates method \cite{20, 21} to represent a point $p$ inside a cage $C_i$ as a linear combination of the cage vertices $\{v_j^C_i\}_{j \in N^C_i}$:

$$p = \sum_{j \in N^C_i} \varphi_j(p)v_j^C_i, \quad \sum_{j \in N^C_i} \varphi_j(p) = 1$$

where $\varphi_j(\cdot)$ are the mean-value coordinates of $p$ inside $C_i$; $N^C_i$ is the vertex number of $C_i$. This equation holds when $C_i$ is deformed.

### 4 Construction of chunk cages

#### 4.1 The segment mesh and the chunk mesh

In this paper we assume that the skeleton of an object model is given. If not, a skeleton for an object model can be either created automatically by skeletonization method \cite{1, 3, 31} or created manually using commercial software \cite{26}.

A model is first decomposed into body segment meshes according to the syntax of bones, as shown in Fig. 2(b). Although we could build one cage for each segment and perform cage-based deformation on it, such cages are bulky and details can be easily lost in the deformation. Instead, we choose to further decompose segments into near-rigid chunks. We then build the chunk cage as the minimal convex hull of each chunk. By near-rigid we mean that vertices belonging to the same chunk tend to experience similar transformation during model deformation. Anatomically, skin or muscle attached to the same bone turns to have similar transformation during model deformation. Anatomically, skin or muscle attached to the same bone turns to have similar transformation during model deformation.

Decomposing an object model into segments is done in a way quite similar to that proposed in \cite{24, 39}. For each joint which links exactly two bones, a plane bisecting the joint angle is used to cut the joint and the mesh into several segments. For joints linking more than two bones, vertices around it are grouped by associating each vertex to the nearest bone. We call the averaging center of each subset, a skeletal point. $p_k$ lies on bone $B_i$, the endpoints of the bone

$$p_k = r_k J_{i,0} + (1-r_k) J_{i,1}$$

where $r_k = \|p_k - J_{i,0}\|/\|J_{i,1} - J_{i,0}\|$, $r_k$ is called the length ratio of $p_k$ on bone $B_i$. This relationship holds when bone $B_i$ undertakes rigid motion.

The skeletal points keep the spatial relative position between the chunk cage and the skeleton as would be detailed in Sect. 5.1. As shown in Fig. 2(c), for clarity, each vertex is linked with its corresponding skeletal point by a line segment.

#### 4.2 The chunk cage

A chunk cage, denoted as $C_k$, is created for each chunk mesh using minimal convex hull algorithms \cite{2, 4}, as shown in Fig. 3(a). Cage vertices form a subset of the set of chunk vertices. For each chunk vertex that is not a cage vertex, it must lies inside the cage and thus can be represented as a linear combination of the cage vertices according to (5).

Generating chunk cages in such a way has several merits. Each chunk cage governs a chunk, which makes it flexible in tackling local deformation. With all the chunk cages working together, the whole object model can be deformed effectively. Since the total number of chunk vertices is only a portion of that of the original mesh, the energy minimization can be preformed more efficiently.
To keep the local details of each chunk cage during model deformation, we define the Laplacian energy of chunk cages as

$$E_L = \sum_{j=0}^{N_c} \sum_{i \in N^C_j} \| L(v_j^C) - \delta_j^C \|^2$$

(7)

where \( N_c \) is the number of chunk cages, \( N^C_j \) is the number of vertices on \( j \)th chunk cage.

### 4.3 Linkage between chunk cages

Since each mesh vertex belongs to only one chunk, there is no vertex shared by two chunks. Thus the chunks are disconnected and gap exists between every two adjacent chunks, and chunk cages as well, as shown in Fig. 3(a). Actually, some edges of the object model fill in these gaps. As shown in Fig. 3(b), the line segments between every two adjacent chunk cages are such kind of edges. These edges serve as dampers to smooth the transition of the deformation difference between every two adjacent chunk cages, so they are named **damping edges**.

Suppose the vertices on the damping edges form the set \( \{v_i\}_D \). By keeping the Laplacian deformation energy on \( \{v_i\}_D \) as small as possible, the deformation on one chunk cage will be transferred to its neighbors. The Laplacian deformation energy on \( \{v_i\}_D \) can be expressed as

$$E_{LD} = \sum_{v_i \in \{v_i\}_D} \| L(v_i') - \delta_i' \|^2$$

(8)

\( E_{LD} \) is used to reduce the deformation difference between adjacent chunk cages. As a result, the object model would be deformed smoothly while keeping its local details well.

### 5 Skeleton-driven deformation

#### 5.1 Skinning

As described in Sect. 4, each cage \( C_k \) has a skeletal point \( p_k \). Therefore, the candidate position of the deformed \( p_k \) can be derived either from the deformed \( C_k \) by using (5), or from interpolating two bone endpoints using (6). The difference between these two candidates is measured by **skeletal point energy** represented as follows.

$$E_{sktl} = \sum \left\| r_k J_{i,0}' + (1-r_k) J_{i,1}' - \sum_{j \in N^C_k} \varphi_j(p_k) v_j^{C_k} \right\|^2$$

(9)

where \( J_{i,0}' \) and \( J_{i,1}' \) are the positions of the two endpoints on bone \( B_i \), \( v_j^{C_k} \) is the coordinate of the deformed vertex \( v_j \) on cage \( C_k \). \( E_{sktl} \) is to be minimized in the deformation.

Equation (9) sets up the relationship between the cage mesh and the skeleton. Rather than binding the mesh surface onto the skeleton explicitly and manually as in the traditional skinning technique, our approach does the binding automatically and implicitly, which greatly reduces the workload for artists.

#### 5.2 Cage constraints along three orthogonal directions

The mean-value coordinates of a point in a chunk cage is actually the coordinates in the cage space. They are invariant under affine transformation or isotropic scaling. Therefore, if we deform an object model by only minimizing the energy of \( E_L, E_{LD} \) and \( E_{sktl} \), an undesirable volume change might occur (see experiment in Sect. 6). To address this issue, sectional point constraints are introduced into the deformation frame.

Each chunk cage \( C_k \) intersects a corresponding bone at two points, or only one point if the cage is enclosing one endpoint of the bone. An intersection point, say \( p_i \), must lie inside a triangle, say \( f_j(v_{j,0}, v_{j,1}, v_{j,2}) \) on \( C_k \), then \( p_i \) can be expressed as the linear combination of three vertex positions. On the other hand, \( p_i \) can be interpolated from two endpoints of the bone. These two results are expected to coincide. Their distance is defined as an energy function,

$$E_{pos1} = \sum_i \left\| \sum_{m=0}^{2} w_m v_{j,m}' - (r_i J_{n,0}' + (1-r_i) J_{n,1}') \right\|^2$$

(10)

where \( J_{n,0}' \) and \( J_{n,1}' \) are the positions of the two endpoints on the deformed bone \( B_n \), respectively. \( v_{j,m}' \) is the position of the deformed \( v_{j,m}, w_m \) is the mth division of the barycentric coordinate of \( p_i \) in \( f_j \). \( (r_i J_{n,0}' + (1-r_i) J_{n,1}') \) can be
Anatomically, skin shape around joints changes more evidently than skin shape around the middle of bones, therefore, the value of $\varsigma_i$ and $\xi_k$ is set decreasingly from the midpoint to the two joints on each bone, which makes chunk cages near the middle of the bone more rigid than those near the joints. Thus, chunk cages near the joints can be deformed more flexibly. $\varsigma_i$ and $\xi_k$ are set as

$$\varsigma_i = e^{-3[0.5 - r_i]} \quad (12)$$

where $r_i$ is the length ratio of $i$th length-wise sectional point on its relevant bone.

$$\xi_k = e^{-3[0.5 - r_k]} \quad (13)$$

where $r_k$ is the length rate of the skeletal point $p_k$ corresponding to chunk cage $C_k$ on its relevant bone. The influence of setting $\varsigma_i$ and $\xi_k$ with different values on the deformation results would be analyzed in Sect. 6.

### 5.3 Deformation computing

Putting all the above introduced energies together, we deform the object model by

$$\text{min}(E), \quad E = \lambda_1 E_{LD} + \lambda_2 E_L + \lambda_3 E_{sklt} + \lambda_4 (E_{pos1} + E_{pos2}) \quad (14)$$

where the coefficients $\lambda_i$ ($i = 1, 2, 3, 4$) are used to balance the importance of the four terms, e.g. $\lambda_1$ is used to control the rigidity of the local region around the damping edges. A bigger $\lambda_1$ preserves more local details around the damping edges in the deformation.

Equation (14) is used to deform the chunk cages, after that the shape of the deformed object model is reconstructed by using (5). After a plenty of experiments, we have found the following parameters setting is appropriate: $\lambda_1 = 3.0$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$, $\lambda_4 = 2.0$. The influence of these parameters on the deformation results will be analyzed in Sect. 6.

Skeleton motion data such as BVH files [8] can be applied to a given model with our technique. Such data are usually created by motion capture equipment and are widely used in the computer animation community. From these data, the joint positions, the rotation matrix on each bone can be parsed. Therefore, the target positions $(r_0 J_{i,0} + (1 - r_0) J_{i,1})$ in (9) and the target positions $(r_0 J_{i,0} + (1 - r_0) J_{i,1})$ in (10) can be easily updated. For updating the target positions $(p_k' + T_k (q_{k,m} - p_k))$ in (11), $T_k$ is assigned to be the transformation matrix on the bone containing $p_k$. Similarly, we assign $T_j$ in (3) to be the transformation matrix on the relevant bone of the vertex. The influence of the local transformation matrix on the deformation results will be analyzed in Sect. 6.
5.4 Numerical solution

Equation (14) is a quadratic function of the new vertex positions on the chunk cages. These new positions could be computed by setting the gradient of E with respective to each new vertex position to be zero. Therefore, (14) can be converted into a linear equation system, as

\[ Ax = b \]

(15)

where \( x \) is the vector for \( m \) cage vertices, \( b \) is a known value vector deduced from (14), \( A \) is an \( n \times m \) coefficient matrix. In our experiments, \( m = 50\% \) of \( N_{V}^{M} \), the vertex number of the object model, \( n = m + s_{1} + s_{2} + s_{3} + s_{4} \) where \( s_{1} \) is the number of vertices of the damping edges, \( s_{2} \) is the number of skeletal points, \( s_{3} \) and \( s_{4} \) are the number of length-wise and width-wise sectional points, respectively. \( m + s_{1} \) is smaller than \( N_{V}^{M} \), since the vertices on the chunk cage and the vertices on the damping edges are the subsets of the vertices on the object model. In our experiments, \( (m + s_{1}) \) is about 75% of \( N_{V}^{M} \). Each chunk cage has one skeletal point, two width-wise sectional points and at most two length-wise sectional points. On average each body segment is subdivided into three chunks, so we have \((s_{2} + s_{3} + s_{4}) \approx 15s_{B}\), where \( s_{B} \) is the number of bones of the skeleton, which is about 20. Note that a 3D character is usually composed with thousands of vertices, in such a case \( 15s_{B} \) is smaller than 5% of \( N_{V}^{M} \). Therefore \( n \) is about 80% of \( N_{V}^{M} \) and \( n \times m \approx (0.8N_{V}^{M}) \times (0.5N_{V}^{M}) \).

For the first \( m \) rows in \( A \), each row is deduced from the Laplacian operator on a certain chunk cage vertex, and the number of nonzeros in the row equals the number of adjacent edges incident to the vertex plus one. For the next \( s_{1} \) rows, each corresponds to the Laplacian operation on a certain object model vertex, which can be further represented by the chunk cage vertices by using (5). For the next \( s_{2} \) rows, each corresponds to a skeletal point, and the number of nonzeros equals the vertex number of the relative chunk cage. For the last \((s_{3} + s_{4})\) rows, each corresponds to a sectional point that is represented by 3 vertices on a chunk cage face. Therefore, the matrix \( A \) is a sparse matrix. The above over-determined system is solved by

\[ x = (A^T A)^{-1} A^T b \]

(16)

Note that while driving the object model by skeleton motion data, only the right-hand-side vector \( b \) varies from frame to frame, whereas the matrix \( A \) is constant. Thus, \((A^T A)^{-1} A^T \) is also constant and is computed only once. Therefore, solving the minimization problem turns into a matrix-vector multiplication operation.

6 Experimental results and analysis

Each energy term in the right-hand side of (14) has its influence on the deformation result. To evaluate their influence on the deformation result, we disable each of them, one at a time, from the system. The consequential deformation results are listed in Fig. 5. Figure 5(a) shows a human model in its initial pose; Fig. 5(b), (c), (d), (e) show the deformation results by configuring (14) with different coefficients. Comparing Fig. 5(b) with Fig. 5(c), (d), (e), we have the following conclusions:

1. Energy term \( E_{LD} \) in (14) smoothes the transition of the deformation between two adjacent cages. Without this term, undesirable deformation would be prone to occur, especially near the joints, as shown the deformation result in the region near the left knee in Fig. 5(c).
2. Energy term \( E_{sklt} \) in (14) constrains each chunk cage being deformed around its bone. Without this term, the chunk cage might deviate from the bone, as shown on the left crus in Fig. 5(d).
3. Energy term \( E_{pos} \) in (14) constrains the volumetric deformation. Without this term, serious volume change might occur, as shown on the left leg in Fig. 5(e).

Coefficients \( \varsigma_{i} \) in (10) and \( \xi_{k} \) in (11) control the cage rigidity. In Fig. 5(b) we set \( \varsigma_{i} \) and \( \xi_{k} \) according to (12) and (13),...
respectively, as a result, the rigidity of the cages changes smoothly along each bone and produce pleasing deformation results. However, if we set $\varsigma_i$ and $\xi_k$ to be a constant value, undesirable deformation result occurs, e.g. when we set $\varsigma_i = 1.0$ and $\xi_k = 1.0$ in Fig. 6, self-intersection occurs in the region near the left knee.

In our approach, every cage vertex assumes the transformation of a single bone. A common sense is that vertices close to a joint should be influenced by multiple bones sharing the joint so that they deform smoothly. However, the tunable rigidity of the cages and the local detail preservation attribute of the Laplacian deformation make this possible. For evaluation purpose, we use the technique of As-Rigid-As-Possible (ARAP) [34] to optimize the deformation result of Fig. 5(b), and the result is shown in Fig. 7. The ARAP technique optimizes the local transformation on vertices by aiming at a smooth deformation transition among adjacent vertices. By comparing Fig. 5(b) with Fig. 7, we can find that the deformation results are very close in shape.

Figure 8 is used to test the performance of our approach on character models with various mesh resolution. The vertex numbers on Fig. 8(a), (c), and (d) are 17 000, 950 and 750, respectively. These models are obtained by refining and simplifying the model in Fig. 8(b), which has 6 499 vertices. As shown in the figures, our approach performs well both on high resolution character models and low resolution character models.

A video demo has been supplemented, in which one female model, one male model and one suit of garment model are animated by a sequence of skeleton motion. Figure 9 shows six frames extracted from the demo.

Anatomically, when a joint bends, the skin around the joint will squeeze. In our experiment, when the pose of the skeleton changes extremely, self-intersection of the object model would happen, as shown in Fig. 10, if no additional treatment is made. Such a problem can be tackled by detecting the self-intersection first, and then simulate the squeeze effects in the specific regions. Since the model surface is inside a series of cages, if there is no intersection between two cages, the chunk meshes would not intersect with each other; therefore we can use these cages to find out those chunk meshes which are likely to collide with each other before the squeezing simulation, which could reduce the computation cost. We would like to develop this technique in the future.

A convincing animation result is ensured only if the coefficient values of the energy function are set appropriately,
including $\xi_i$ in (10), and $\xi_k$ in (11) and $\lambda_i$ ($i = 1, 2, 3, 4$) in (14). We prefer to expose these parameters to the user in the future, and develop a tool for interactively tuning the coefficients. If any of these parameters is changed by the user, either at the starting stage or in the middle of the frame composing, matrix $A$ needs to be re-computed and so does $(A^TA)^{-1}A^T$. Once this is done, the animation frames will be synthesized in real time. We would also like to develop a system to automatically evaluate the deformation results. When the system detects undesirable deformation in a certain frame, the user is prompted to change the coefficients to produce more convincing result.

We have implemented our approach using C++ on a computer with Intel(R) Core(TM)2 Quad CPU Q8300 @2.50 GHz, 3.00GRAM. The vertex numbers of the female human model, the male human model (other than the model in Fig. 8(a), (c), (d)) and the garment model are 6 574, 6 449 and 3 158, respectively. The total computational time of model segmentation, chunk cages creation and skinning on the three models is 1 438 ms, 1 328 ms, and 602 ms, respectively. Such processing on Fig. 8(a), (c), (d) takes 7 765 ms, 47 ms and 31 ms, respectively. This is done only once for each model. The computational time for deforming the object model into a proper pose by (16) is under 5 ms once for each model. The computational time for deforming the object model and the skeletal points with the chunk cages. The model deformation is achieved by minimizing the object model into a proper pose by (16) is under 5 ms once for each model. The computational time for deforming the object model and the skeletal points with the chunk cages. The model deformation is achieved by minimizing the energy function which contains the terms of local details preservation, deformation smoothness and volume change prevention. The energy minimization problem is linearized and only a matrix-vector multiplication is perform for each frame of animation. Our approach can produce plausible deformation results for articulated characters.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Nos. 60903145, 6107105), Beijing Natural Science Foundation (No. 4102062), and the Open Project Program of the Engineering Research Center of Digitized Textile & Fashion Technology, Ministry of Education, Donghua University. We would like to thank Nils Hasler for providing the male model.

References


Jituo Li is currently an assistant professor of the Department of Mechanical Engineering, Zhejiang University, China. He received his BS degree from Central South University of China in 2000 and Ph.D. degree from Zhejiang University in 2006. Before moving back to Zhejiang University in June 2010, he was with the Institute of Automation, Chinese Academy of Sciences. His research interests include mesh editing, modeling and animation.

Guodong Lu is currently a Professor and the Deputy Director of Engineering and Computer Graphics Institute at Zhejiang University. He received his B.Sc., M.Sc. and Ph.D. from Zhejiang University in 1983, 1990 and 2000, respectively. His research interests include CAD/CAM in soft-products industry, 3D reconstruction from 2D engineering drawings.

Juntao Ye was awarded his M.Sc. in Applied Mathematics from Institute of Computational Mathematics and Sci/Eng Computing, Chinese Academy of Sciences in 2000, and his Ph.D. in Computer Science from University of Western Ontario, Canada, in 2005. He was a coder for a video game studio in Ontario from 2005 to 2007. He is currently an associate professor at Institute of Automation, Chinese Academy of Sciences. His research interests include cloth simulation and augmented reality.